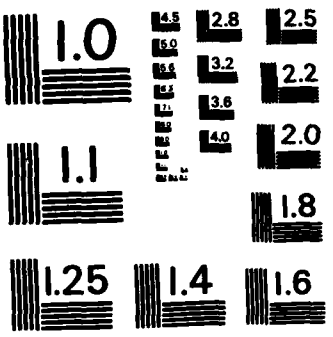
AD-A159 489 SSMS - A SECURE SOFTWARE MANAGEMENT SYSTEM(U) ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (ENGLAND) G D WHITAKER NOV 84 RSRE-MEMO-3777 DRIC-BR-96554 1/1

UNCLASSIFIED F/G 9/2 NL

END

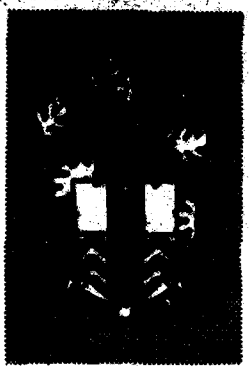
FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

(3)



RSRE
MEMORANDUM No. 3777

ROYAL SIGNALS & RADAR ESTABLISHMENT

SSMS – A SECURE SOFTWARE MANAGEMENT SYSTEM

Author: G D Whitaker

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

(A)

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3777

TITLE: SSMS - A SECURE SOFTWARE MANAGEMENT SYSTEM

AUTHOR: G D Whitaker

DATE: November 1984

SUMMARY

British document

This ~~memorandum~~ describes a system for managing computer software, documentation etc in a secure manner. The system relies on a central database, a diary of system updates and a number of utility programs. These are all described in the text together with a method of running the system which protects the integrity of the software being managed.

KEYWORDS

Software management
security
database
ADAM

SSMS - A SECURE SOFTWARE MANAGEMENT SYSTEM

G D Whitaker

List Of Contents

1 Introduction

The Central Computing Facility at RSRE is currently provided primarily by an ICL 1906S mainframe computer running under the George 3 operating system. A 1900 series machine has been in use since 1971 and the software supporting the central service has been developed and improved continually ever since. A number of changes within Computing Service made it prudent to reconsider how this software was managed.

The most obvious reason for change was that after ten years of development, the system was becoming too extensive for an individual to be aware of all the facilities available. This was aggravated by the loss of several key staff who had been involved with the design and growth of the software for many years. Experience and knowledge on this scale is very difficult to pass on, even if new staff had been available to cover the gaps. Reductions in staffing levels in "support" areas made the situation even more critical especially when large numbers of users were requesting an increase in technical support which left little time for the remaining staff to become familiar with the system.

The 1906S will be replaced before the spring of 1987. A new system will be installed and will come with some support software. It will be important to determine, as soon as possible, how this software compares with the existing facilities. In particular, are there any gaps in the new facilities which will require new software to be purchased or developed? This assessment will require a complete knowledge of the existing facilities to be available.

The two factors outlined above indicated a requirement for a directory of the existing software. This directory would be required to contain, as a minimum, the name of the item, the type of the item (eg macro, segment, documentation etc), the purpose of the item and its relationship with other items. As there is a small, but nevertheless significant, amount of continued development and modification of the existing facilities, it would be vital to keep this directory up to date. In addition, it would be useful to automate the current manual method of making and recording changes made to system software.

The final factor influencing this change of approach was the advent of a Multi-Level Secure Service on the 1906S. Before this increase in security it was just about acceptable to update system software in an ad hoc manner. This ad hoc method would not be good enough to form part of the new Secure Operating Procedures.

2 The Old Method

The manual method of curing a fault or improving system software was to:

1) Find the source text
2) Make the modification and test in a development username
3) Ask the operations manager to update the live software
4) Make a note of the change in the software log

This non-automatic, informal method of working was fraught with possibilities for deliberate or accidental sabotage. Fortunately, only a few people were involved and they were fairly conscientious in following some rough guidelines. Also, no person, as far as we are aware, had the inclination, knowledge and opportunity to inflict deliberate damage on the central system. We could not rely on this always being the case.

The system relied on human intervention at all levels and could easily fall down if one job was forgotten or run in the wrong order. It also failed to keep everyone concerned in running system software informed of changes and relied heavily on a hand written entry in the software log. This entry could easily be forgotten or could be so cryptic that not even the originator could remember the significance of some notes. The date and time were put in manually and so could be considerably different from the actual time the change took place. It was too easy to delay an entry because "I will do it as soon as I have done ..... " or "I will put the entry in when all these quick changes have finished and the system is stable again.".

System software is used from a number of system usernames. Some of this software (eg George macro-commands) is stored and used in a readable form and so modifications can be based on the current 'live' software. Other items (eg program segments) cannot be interpreted from their live form and it is necessary to return to the original source text. This source text was not stored in any formal manner and it was left to the originator to safeguard the text.

The people concerned with modifying system software usually stored the text in one of a number of system development usernames or on a magnetic tape owned by those usernames. There was no guarantee, however, that the stored text was consistent with the live software or even that the source text could still be found! It was also possible for two people to work on the same item of software independently. Each would believe that their version of the source text was the most recent. Thus, bugs which had been cured by one programmer could be reintroduced when another bug was cured or the software improved by another.

All too often software is updated on other computers without any records being kept of the changes and no checking that modifications are legitimate. Also, it is rare for source text to be stored centrally. These other machines are generally run by a number of individuals rather than one section and so coordination is even more difficult to achieve.

The system development usernames were not regarded as secure. It is true that very little system software is classified but, since much of it is relied upon by secure users, it is vital that this software is not modified except by authorised people in an authorised and monitored manner. The live software is afforded this level of protection but the development software, including the source text of system software, was not similarly protected.

A major limitation of this manual system was that it did not provide any feasible method by which the operations manager could check that the system was correct. It was possible to carry out 'spot checks' on particular files but the effort required to confirm that, for instance, there were no extra files in the system, no files missing from the system and that all the files had the correct security levels and access rights, was too daunting. This was assuming that the 'correct' levels etc. could always be remembered.

3 The New System

The new system is largely automatic, easy to use and difficult to get wrong. Central to this system are two data files which are stored in a secure system username. One file, "sysdiary", is a character file which contains an entry for every change made to system software. The second file, "sysindex", is a binary (random access) file which contains an entry for each item of system software, system documentation, data files etc. These two files will be described in more detail later.

It is only possible to append information to the end of sysdiary and this is only permitted from certain secure usernames. Thus, sysdiary provides a secure and complete record of system changes as and when they occur and, because it is updated automatically, there is no opportunity for forgetfulness.

There is only one utility which can write to sysindex. This utility is called "syschange" and may be run only from the same secure usernames.

The username which owns sysindex and sysdiary is used to store copies of the source text of all system software. A utility, "sysdump", has been developed which ensures that all new or modified files in that username are also archived on magnetic tape. This archive utility is in addition to the standard George dumping mechanism.

All system development work is carried out in one username. Read access to all system source text has been granted to this username but no modifications are permitted. In addition, this username has had its security level raised for increased protection of the software during development (but not to the same level as that of the software storage username because it was felt that that level was too restrictive for large scale program development).

A utility is provided which can be used to check that the sysindex ('correct') view of the system is matched by the actual files in the filestore. This utility identifies any extra or missing files and also highlights any discrepancies in security levels or access rights etc.

The sequence of events for updating an item of system software is as follows:

1) The current source text is copied from the software storage username to the system development username. It would be useful to use a utility for this operation which could quiz sysindex to find the name of the file, make a note that the copy has taken place and prevent two people from working on the same software independently.

2) The software is modified in the medium security system development username.

3) When the modifications are complete and have been tested, the operations manager runs a utility to
   a) copy the source text from the development username into the storage username
   b) update the live software
   c) produce two line printer listings of the software (one for the programmer and one for the master listings held by the operations manager)
   d) note the update in sysdiary
   e) note the update in sysindex

4) The development software is erased leaving the only copy of the source text in the secure software storage username.

4 The System Diary

This file is an automatic software log. The diary contains an entry for every change made to system software together with the date and time of that change. All this information is appended to the end of the file automatically and so provides a precise account of all system changes. In general it is simpler and more reliable to consult the system diary to discover what changes have been made rather than rely on a manual search through the handwritten log. In theory, it would be possible to dispense with the old log. In practice, it is useful to maintain the old software log (by hand) in parallel with the the system diary. The handwritten log is always available even when the computer itself is not and it contains a good approximation to the actual changes made to the system.

The system diary is a character file and so may be listed using standard George commands. It is rare for the system diary to be listed in its entirety. More frequently it is only changes made to a certain item of software that are required or maybe changes made between two dates or times. Consequently it is usual to interrogate the system diary by means of a special utility, "diaryprint".

The diaryprint utility prompts for the fields the user is interested in. The possible fields currently include the name of the item, the type of the item (eg macro, documentation etc) or the dates delimiting the days required. Output from diaryprint can be to the terminal, to a file or to the line printer and includes the entries in the diary which match the fields specified. This means that it is possible to identify all changes made to a given item or to list all macros which have been changed during the first two weeks in September 1984 etc.

5 The System Index

This file is a complete index of all system software and includes George macro-commands, program segments, data files, magnetic tapes and documentation. The list of entry types is not exhaustive and it is possible to introduce new types. All items in the index consist of a number of fields. Some of these fields, such as the name of the entry, are simply strings of characters, others, such as entry type, are stored internally as integers (although a user of any of the utilities accessing the index will not be aware of this) and others, such as 'other items which this entry uses', are references to other items.

A binary (direct access) file is more appropriate than a character file for this sort of application. The only advantage of a character file would be the availability of standard George editors and listing commands to manipulate and read the data stored in the index. Against that, forcing access to the index to be by means of specially written utilities provides more flexibility and greater security.

The security is increased because any potential attacker would have to write suitable software and could not simply take advantage of editors and the like. Even if a would be attacker were capable of writing such a program and could find out the internal format, the effort involved might deter an opportunist attack. Also, authorised users are unlikely to be tempted to by-pass the standard utilities in order to effect a quick patch and so there is less possibility of making a mistake such as not matching a cross-reference.

The flexibility comes from the ease of maintaining cross-references and the ability to hide implementation details such as 'entry types are stored as integers'. Since a program has to be run to write to or modify an entry, and a program run to list the entries, these programs can hide such details of implementation as they see fit. Thus users are presented only with information which is meaningful to them but the program (there is only one which writes to the index) can ensure that the data is internally consistent and stored efficiently.

6 Programming For The System Index

It can be a tedious job to write the software necessary for manipulating a database such as sysindex, but fortunately such software was being developed elsewhere. Although at that time it was still experimental and only provided basic functions, the ADAM [1] package appeared to offer some very useful facilities. This package was acquired at an early stage and both ADAM and SSMS developed in parallel. Consequently some of the features which were written as part of the SSMS system were of more general appeal and are now available in ADAM.

ADAM takes care of the intricacies involved in handling binary files and provides the programmer with a safe, yet versatile, method of storing data. Used in this way, a programmer is free to concentrate on the data itself rather than on the data handling aspects. The philosophy is that data is presented to an ADAM procedure for safe keeping and the program is given a capability (Capability: an ability to do something - in this case, retrieve the data) for the later retrieval of that data. It is not possible to forge such a capability but they may themselves be stored within an ADAM type file. ADAM also takes care to safeguard the data in the event of a computer failure during updating.

Conceptually, the system index consists of one capability - a root to the whole database. This may be obtained by calling a suitable ADAM procedure. The format of the database is then at the discretion of the programs which access it. It is a reasonably simple job to restructure the database completely although, so far, this has not been necessary. The point to note though is that it is possible to add new structures to the database or modify the structures already present with relatively little effort and without a user of the SSMS system being aware of the change. What follows, therefore, is merely a description of the index as it presently stands. The methods employed highlight the concepts involved and will serve to illustrate the technique.

The root capability may be used to read an array of capabilities. Each element of this array is a capability (called a disc pointer by ADAM) to access one entry in the index. Each entry contains some, but not necessarily all, of the fields listed below. An entry need not contain all the fields for two reasons. Firstly, a field may not be applicable to an entry or the information forming that field may not be known for the entry. Secondly, new fields may be introduced without the necessity to enter that field in all the entries already typed in. The mechanism by which this works will be described later.

The fields forming an entry in the index include:

Name	(by which the software is commonly known)
Type	(an integer specifying whether macro, documentation etc)
Source	(name of file containing the source text)
Location	(of the actual item)
Purpose	(one line description)
Date	(of last modification) } These can be cross-referenced
Time	(of last modification) } in the system diary
Used by	(other entries which rely on this item)
Uses	(other entities which this item relies on)
Owner	(last person known to be responsible for this item)
Traps	(giving the access rights - traps - of George users)
Comment	(any special comment)
Classification	(level required to read the file)
Integrity	(level required to modify the file)
Reason	(for the last modification - same as in system diary)
Log access	(whether or not access to the file is logged)
Archive tape	(magnetic tape containing the archived copy)

The ADAM package recognises arrays of characters, arrays of integers, and arrays of disc pointers. It is also possible to have arrays containing a mix of these modes. A glance at the above fields indicates that it should be possible to store all the required information using these modes provided a translation facility is written to convert, for example, a type into an integer for storage in the file and the integer back into a type for displaying to a user. This was the initial approach adopted but soon had to be abandoned when it was realised that for every field presented to the ADAM interface, a new block was taken in the file. Most fields only required a small number of words for storage and so a large proportion of the available space was being wasted. This could not be tolerated for the large number (many hundreds) of entries stored in the index.

The solution adopted to make better use of the file is now the approach recommended in the ADAM report. A simple code is employed which enables the fields to be concatenated. This method has been taken to the extreme and all the fields described above are concatenated into one, coded string. Low level procedures are provided in the software to add fields of various types into an entry. This makes it a relatively simple task to introduce a new field into the database.

A field is comprised of three parts, a unique field sort number, the size of the data and the data itself. Since all programs which access the system index do so by means of the same central module, there is no scope for inconsistencies in the unique field sort number. To construct one string of characters containing all three parts, a certain amount of translation is necessary. This translation is carried out at a very low level within the common module.

A field sort number is simply the character representation of that number. This provides scope for 64 different sorts which, considering only 17 have been required so far, should be ample for this application. The size of the data is stored as the four character decimal representation of the size, in characters, of that data. Although by no means an optimum packing mechanism, this permits field data containing 9999 characters which is in excess of anything yet required and only occupies four characters of space.

The SSMS requires to store arrays of characters, integers and references in sysindex and these can be made into complete fields as follows:

Arrays of characters simply have the field type and data size added to compose the complete field.

Integers (which only occur as the representation of a limited set such as type or classification etc) are converted into the four character decimal representation of that number using the same procedure as that used for the size of data component above.

References to other items are stored in the same way as integers! The integer is the position, within the array of entries, holding the item to which this field refers. Current estimates indicate that the file will become full long before 9999 items are entered. It is unlikely that this application will overflow the file but the situation is being monitored. Should further files be required, then the most significant digit could be used to identify which file is being referenced. Alternatively, ADAM now permits 'slave files' and this may be worth pursuing.

To introduce a new field type it is necessary to modify one segment which is central to all the programs accessing the index. The new type is allocated a number not used by any other type. The form the data is to take (array of characters, integer etc) is established and the user interface written. It is not necessary to add this field to all the entries already in the index.

The whole system relies on the fact that every entry is unique. If this were not so then it would be possible to update the first of two identical entries but to access the second when looking for that entry. The system insists that all name, type combinations are unique. Thus by specifying a name and a type the system can locate the unique entry or alternatively know that the entry is not in the index. It is not possible to make an entry in the index without specifying a name and a type as a minimum. In addition to this a date and time will be entered automatically and there is no method of avoiding this.

7 Utility Programs

There are a number of utility programs providing the overall Secure Software Management System. The diaryprint utility has already been described for listing entries from the system diary. Several utilities dealing with the system index are described below.

The indexprint utility is similar to diaryprint in that it invites the user to select which fields are to be matched and then lists the matching entries. Possible fields for matching purposes are the name, type and owner fields. The utility then proceeds to enquire about the form the output is to take. In particular, the user is given a choice as to which fields are to be printed and whether the list should be ordered alphabetically.

The indexprint program actually uses a lower level "indexlist" segment. This segment deals with access to sysindex and with sorting the entries but allows the driving program (indexprint) to select the entries for listing, to determine the algorithm for sorting and to print the entries as it sees fit. It is a relatively simple task to write an alternative indexprint (or modify indexprint itself) if, for instance, it was desirable to print the entries in alphabetical order of owners or other similar variations on the theme.

The system index should be an accurate picture of the state of the system files within the filestore. To guarantee that this is in fact the case, a utility, "syscheck", is provided to compare the actual state of the filestore with the sysindex view of the situation. This uitility not only checks for files which have no entry (and entries with no corresponding file), but also checks that some of the fields within sysindex are also correct. In particular, the classification, integrity and traps fields are currently checked. All inconsistencies are noted and listed for manual investigation.

The "syscheck" utility also proved useful when the index was first being set up. At that time it was reasonable to assume that the filestore was likely to be a more accurate picture of the 'correct' situation than the index and so it was a fairly automatic process to update the index to match the filestore and later carry out a manual check that this was indeed the 'correct' situation. A number of files were found to have incorrect trap settings which were only present for historical reasons. These loopholes into the system were soon closed.

The George operating system provides a very safe filestore. All files are well protected by a regular dumping system and it would be possible to keep all system files in the George filestore. This could lead to a very large number of previous versions of software in the filestore which would probably never be required again. In such situations it is good practice to archive the files on private magnetic tapes. The utility "sysdump" arranges for any files modified within a particular month to be archived. Ideally, this utility should be run every month to archive any files created or modified during the previous month. Any file which has been superseded may then be erased from the filestore. In practice files would only be erased after a sufficiently long period had elapsed to be confident that the new software was an improvement over the old. As an extra safety measure a copy of the archive tapes is also maintained.

The sysdump utility also serves as an extra security check. A list of the files being archived is produced. This list contains the names of all files which were created or modified during the month in question. This is then compared with both the diary and the index to confirm that an entry corresponding to that file has been made. Runs of sysdump and syscheck are also carried out at irregular times thus catching any file created and erased between the regular runs. This is in addition to the George logging mechanisms.

8 Conclusions

The initial version of the Secure Software Management System was installed in the summer of 1983. This primitive version handled all changes to system software for the next 9 months until the current version (described here) took over. It took several months to identify and enter most of the existing software but by the spring of 1984 there was an entry for every item of system software.

During the summer of 1984, the system was completed. Several utilities were improved (mainly as a result of feedback from users of the system during the year) and the remaining utilities written. Since then the new system has been responsible for all the system software relied upon by the Central Computing Service at RSRE.

A system such as this can only achieve its aim - to provide a secure system - if it is simple and convenient to use. If it is difficult to understand or unreliable or in any other way inconvenient, then means will be found of bypassing the system. The operations staff at RSRE have taken to the system with enthusiasm (after the tedious job of entering the backlog in the first few months). I think this, more than any description of the software itself, indicates the simplicity of the user interface and the convenience of the overall system.

If effort were still available, there are several areas which might be improved. The weakest link in the system is the connection between the update of the software and the entry in the system index which still behaves as two separate activities and requires some information to be supplied twice. With extra thought and effort, a much better tie up could be achieved.

The user interface was restricted by the necessity to be able to call the update utility from a teleprinter for security reasons. If a secure video terminal were available then all the benefits of interacting with a video screen could be obtained. This would, of course, require some work on software to manage video terminals. If this software were available now, then it would be possible to improve the user interface for those applications which are permitted to access the database from a video terminal.

Perhaps the greatest return for effort invested would be to separate out the implementation independent parts from those parts specific to the SSMS. For instance, procedures which provide a level above ADAM for this type of application and procedures which interact with the user could be identified and be made into distinct packages. These could then be useful to others wishing to use such techniques in other areas.

One of the major strengths of ADAM is its portability. ADAM has already been implemented on the VAX computer. The SSMS software has been written in Algol68 which is now available on a number machines (including the VAX computer) and will be the common language available on most computers at RSRE. It should, therefore, be a simple task to install SSMS on a VAX, or any other computer supporting Algol 68, and thus provide a secure software management system. SSMS could also be implemented on the ICL 1906S replacement computer(s).

9 Acknowledgements

Acknowledgement is due to Malcolm Hopper who has been involved in all stages of this project, from initial discussions, through programming and user feedback to final implementation. I would also like to thank Paulette Cody (currently a student at University College, Cardiff) who worked hard on the programming during her summer vacations, contributed many useful comments and who provided the final impetus to complete the project.

The work of Nancy Davis and the Operations staff in the task of entering the considerable quantities of data and the continued running of the system also deserves recognition.

I would also like to thank Bob Bateman and Susan Bond for reading draft versions of this memorandum and making some useful comments.

10 References

1. ADAM: AN ABSTRACT DATABASE MACHINE - RSRE Report 84007
   N E Peeling, J D Morison, E V Whiting

DOCUMENT CONTROL SHEET

Overall security classification of sheet ..... UNCLASSIFIED .....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	MEMORANDUM 3777	AD-A159489	UNLIMITED

5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location
	ROYAL SIGNALS AND RADAR ESTABLISHMENT

5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title

SSMS - A SECURE SOFTWARE MANAGEMENT SYSTEM

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
WHITAKER, G D				

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement

UNLIMITED

Descriptors (or keywords)

SOFTWARE MANAGEMENT
SECURITY
DATABASE
ADAM

continue on separate piece of paper

Abstract

This memorandum describes a system for managing computer software, documentation etc in a secure manner. The system relies on a central database, a diary of system updates and a number of utility programs. These are all described in the text together with a method of running the system which protects the integrity of the software being managed.

END

FILMED

11-85

DTIC